**REAL-TIME DRIVER ALERTNESS SYSTEM WITH ALARM**

**A**

## Real Time Research/Societal Project Report

***Submitted to***



Jawaharlal Nehru Technological University, Hyderabad

*In partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

By

**MUTHYAMAINA MOUNIKA 22VE1A05A1**

**Under the Guidance**

**of Mrs.P.VIJAYALAKSHMI ASSISTANT PROFESSOR**



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi) Beside InduAranya, Nagole, Hyderabad-500068, Ranga Reddy Dist.

**(2022-2026)**



# SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Real Time Research/societal Project Report on **“Real-Time Driver Alertness System With Alarm”** submitted by **Muthyamaina Mounika** bearing Hall ticket numbers: **22VE1A05A1** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2023-2024 is a record of bonafide work carried out by them under our guidance and Supervision.

.

## Project Coordinator Head of the Department M.SUDHAKAR Dr. U. M. FERNANDES DIMLO

**Assistant Professor HOD & Professor**

## Internal Guide Mrs.P.VIJAYALAKSHMI

**Assistant Professor**

**External Examiner**



# SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I **Muthyamaina Mounika** bearing Hall ticket number: **22VE1A05A1** hereby declare that the Real Time Research/societal Project titled

**REAL-TIME DRIVER ALERTNESS SYSTEM WITH ALARM** done by us under the

guidance of **Mrs.P.VIJAYALAKSHMI, Assistant Professor** which is submitted in the partial fulfillment of the requirement for the award of the B.Tech degree in **Computer Science and Engineering at Sreyas Institute of Engineering and Technology** for Jawaharlal Nehru Technological University, Hyderabad is our original work

**MUTHYAMAINA MOUNIKA 22VE1A05A1**

**ACKNOWLEDGEMENT**

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

I take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mrs.P.Vijayalakshmi, Assistant Professor, Department of Computer Science and Engineering** for his constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **Dr. U. M. Fernandes Dimlo, Head of the Department** and **M.Sudhakar, Project Coordinator** who has been a source of Continuous motivation and support. They had taken time and effort to guide and correct us all through the span of this work.

I owe very much to the **Department Faculty, Principal** and the **Management** who made our term at Sreyas Institute of Engineering and Technology a stepping stone for our career. We treasure every moment we had spent in college.

Last but not the least, our heartiest gratitude to our parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

**MUTHYAMAINA MOUNIKA 22VE1A05A1**

# ABSTRACT

The Real-Time Driver Alertness System with Alarm is a cutting-edge solution designed to enhance driver safety by monitoring and analyzing facial landmarks, specifically focusing on eyelid blinking patterns to accurately assess driver alertness. By leveraging advanced technologies such as dlib for precise facial landmark detection, cv2 for real-time video processing, and yolov5 for robust object detection, this system excels in identifying signs of drowsiness and distractions. One of the key features of this system is its ability to detect common driver distractions, including mobile phone use and eating, which are critical factors contributing to road accidents. By continuously monitoring the driver’s facial expressions and behavior, the system can determine when the driver is becoming drowsy or distracted. Upon detection of such conditions, the system promptly triggers an alarm, alerting the driver to take immediate precautionary measures or rest, thereby preventing potential accidents. The implementation of this system is both user-friendly and efficient. The use of dlib allows for accurate detection of facial landmarks, while cv2 facilitates real-time processing of video feeds from a camera mounted within the vehicle. Yolov5 enhances the system's capability to identify and classify objects within the video frame, ensuring reliable detection of distractions. This proactive approach not only helps in reducing the risk of accidents caused by drowsy or distracted driving but also promotes safer driving habits. The simplicity and effectiveness of the Real-Time Driver Alertness System make it an invaluable tool for enhancing road safety. Its ability to provide timely alerts and promote preventive measures significantly contributes to reducing road incidents and saving lives, making it a crucial addition to modern vehicle safety systems.

**KEYWORDS:** Driver Alertness, Drowsiness Detection, Distraction Detection, Real-time Monitoring, Safety.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** |  | **TABLE OF CONTENTS** | **PAGE NO.** |
| **1** |  | **INTRODUCTION** | **1-6** |
| 1.1 | GENERAL | 1 |
| 1.2 | PROBLEM STATEMENT | 2 |
| 1.3 | EXISTING SYSTEM | 3 |
| 1.3.1 | DRAWBACKS | 4 |
| 1.4 | PROPOSED SYSTEM | 5 |
| 1.4.1 | ADVANTAGES | 6 |
| **2** |  | **LITERATURE SURVEY** | **7-9** |
| 2.1 | TECHNICAL PAPERS | 7 |
| **3** |  | **REQUIREMENTS** | **10-11** |
| 3.1 | GENERAL | 10 |
| 3.2 | HARDWARE REQUIREMENTS | 10 |
| 3.3 | SOFTWARE REQUIREMENTS | 11 |
| **4** |  | **SYSTEM DESIGN** | **12-24** |
| 4.1 | GENERAL | 12 |
| 4.2 | SYSTEM ARCHITECTURE | 14 |
| 4.3 | UML DISGN | 15 |
| 4.4 | USE-CASE DIAGRAM | 16 |
| 4.5 | CLASS DIAGRAM | 18 |
| 4.6 | ACTIVITY DIAGRAM | 21 |

|  |  |  |  |
| --- | --- | --- | --- |
| **5** |  | **TECHNOLOGY DESCRIPTION** | **25-32** |
| 5.1 | WHAT IS PYTHON ? | 25 |
| 5.2 | ADVANTAGES OF PYTHON | 26 |
| 5.3 | LIBRARIES | 29 |
| 5.4 | DISADVANTAGES OF PYTHON | 31 |
| **6** |  | **IMPLEMENTATION** | **33-40** |
| 6.1 | METHODOLOGY | 33 |
| 6.2 | SAMPLE CODE | 35 |
| **7** |  | **TESTING** | **41-42** |
| 7.1 | GENERAL | 41 |
| 7.2 | TYPES OF TESTING | 41 |
| 7.3 | TEST CASES | 42 |
| **8** |  | **RESULTS** | **43** |
| 8.1 | RESULTS SCREENSHOTS | 43 |
| **9** |  | **FUTURE SCOPE** | **44-45** |
| **10** |  | **CONCLUSION** | **46** |
| **11** |  | **REFERENCES** | **47-48** |

|  |  |  |
| --- | --- | --- |
| **FIG. NO/TAB.NO** | **LIST OF FIGURES AND TABLES** | **PAGE NO.** |
| 4.1 | Architecture Diagram | 14 |
| 4.2 | Use Case Diagram | 18 |
| 4.3 | Class Diagram | 21 |
| 4.4 | Activity Diagram | 24 |
| 7.1 | Types of Testing | 41 |
| 7.2 | Test cases table | 42 |

|  |  |  |
| --- | --- | --- |
| **SCREENSHOT. NO** | **LIST OF SCREENSHOTS** | **PAGE. NO** |
| 8.1 | Output | 42 |

# LIST OF SYMBOLS

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO.** | **Name of Symbol** | **Notation** | **Description** |
| 1 | CLASS |  | Represents a collection of similar entities grouped together. |
| 2 | ASSOCIATION |  | Associations represent static relationships between classes. Roles represent the way the two classes see each other. |
| 3 | ACTOR |  | It aggregates several classes into a single class. |
| 4 | RELATION (uses) | *Uses* | Used for additional process communication. |
| 5 | RELATION (extents) |  | Extends relationship is used when one use case is similar to another use case. |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | COMMUNICATION |  | Communication between various use cases. |
| 7 | STATE |  | State of the process |
| 8 | INITIAL STATE |  | Initial state of the object |
| 9 | FINAL STATE |  | Final state of the object |
| 10 | CONTROL FLOW |  | Represents various control flow between the states. |
| 11 | DECISION BOX |  | Represents decision making process from a  constraint |
| 12 | USE CASE |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | COMPONENT |  | Represents physical modules which is a collection of components. |
| 14 | NODE |  | Represents physical modules which are a collection of components. |
| 15 | DATA PROCESS/ STATE |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 16 | EXTERNAL ENTITY |  | Represents external entities such as keyboard,  sensors, etc |
| 17 | TRANSITION |  | Represents communication that occurs between processes. |

|  |  |  |  |
| --- | --- | --- | --- |
| 18 | OBJECT LIFELINE |  | Represents the vertical dimensions that the object  communications. |
| 19 | MESSAGE |  | Represents the message exchanged. |

**CHAPTER 1 INTRODUCTION**

## GENERAL

In today's fast-paced world, ensuring the safety of drivers is paramount. With increasing cases of road accidents attributed to driver fatigue and distractions, there is an urgent need for effective solutions to enhance road safety. The Real-Time Driver Alertness System with Alarm addresses this critical issue by leveraging advanced technologies to monitor and analyze driver behavior in real-time.

This innovative system integrates facial landmarks analysis, object detection, and auditory alerts to create a comprehensive solution for preventing accidents caused by drowsy or distracted driving. By utilizing tools such as dlib for facial landmark detection, OpenCV (cv2) for image processing, and the YOLOv5 model for object detection, the system can accurately assess a driver’s alertness based on eyelid movements and the presence of distracting objects like mobile phones and food items.

The core functionality of the system revolves around detecting signs of drowsiness by analyzing the Eye Aspect Ratio (EAR). When the EAR falls below a certain threshold for a predefined duration, the system triggers an alarm to alert the driver, encouraging them to take precautionary measures. Additionally, the system employs YOLOv5 to identify potential distractions within the vehicle, issuing warnings when necessary to help the driver maintain focus on the road.

Beyond real-time monitoring, the system also logs incidents to a database, providing valuable insights into driver behavior patterns over time. This data can be used to further refine the system's algorithms and contribute to broader road safety initiatives.

In summary, the Real-Time Driver Alertness System with Alarm represents a significant advancement in road safety technology. Its ability to detect and address both drowsiness and distractions in real-time makes it an invaluable tool for reducing accidents and saving lives on the road.

## PROBLEM STATEMENT

Road safety remains a critical concern globally, with driver fatigue and distractions being major contributors to traffic accidents. The traditional methods of ensuring driver alertness, such as periodic breaks and manual checks, are often insufficient and unreliable. This leads to several pressing problems that necessitate the development of advanced technological solutions.

1. Driver Fatigue and Drowsiness: One of the leading causes of road accidents is driver fatigue. Fatigue can significantly impair a driver’s reaction time, decision-making ability, and overall alertness, making it a substantial risk factor for accidents. Current methods to monitor driver fatigue, such as self-reporting or periodic breaks, are not proactive and fail to provide real-time intervention when a driver is about to fall asleep.
2. Distracted Driving: In today’s digital age, distractions such as mobile phones, food, and other objects within the vehicle have become commonplace. These distractions divert the driver’s attention away from the road, increasing the likelihood of accidents. Traditional approaches to minimize distractions, such as laws and regulations, are often difficult to enforce and monitor consistently.
3. Lack of Real-Time Monitoring Systems: Many existing solutions lack the capability to monitor and analyze driver behavior in real-time. This delay in detection and response can result in catastrophic outcomes. There is a need for a system that can provide instantaneous feedback and alerts to the driver to prevent accidents.
4. Insufficient Data Logging and Analysis: Most current systems do not offer robust data logging capabilities to track and analyze driver behavior over time. This data is crucial for understanding patterns, improving system algorithms, and implementing preventive measures effectively.
5. User Acceptance and Ease of Use: Many existing technologies are either too complex to use or intrusive, leading to poor user acceptance. There is a need for a solution that is not only effective but also user-friendly, ensuring that drivers are willing to adopt and use it consistently.

The Real-Time Driver Alertness System with Alarm addresses these problems by combining advanced facial landmark analysis, object detection, and real-time auditory alerts to create a comprehensive and user-friendly solution for enhancing driver safety and reducing road accidents.

6.

## EXISTING SYSTEM

Several advanced systems exist today to address driver fatigue and distraction using various technologies. Here are five notable examples:

1. **Guardian by Seeing Machines**: This system leverages facial recognition and eye-tracking technology to monitor driver behavior. By detecting signs of fatigue and distraction, it can trigger alerts and interventions to prevent accidents. The system continuously analyzes the driver’s eye movements and head position to identify potential risks, ensuring a high level of safety.
2. **EyeSight by Omron**: EyeSight employs camera-based technology to observe driver behavior. It focuses on detecting signs of drowsiness or distraction and issues timely alerts to ensure safe driving. The system's advanced algorithms analyze facial expressions and eye movements, providing a robust solution for maintaining driver alertness.
3. **Driver Attention Monitor by Lexus**: Lexus has equipped its vehicles with a driver attention monitor system that uses a combination of steering inputs and facial recognition to detect signs of fatigue. When the system identifies that the driver is becoming inattentive, it prompts them to take necessary breaks, thus preventing potential accidents caused by drowsiness.
4. **Fatigue Detection System by Bosch**: Bosch's system uses steering angle sensors and various vehicle data points to monitor driver behavior. By analyzing these inputs, it can detect signs of drowsiness and alert the driver to take corrective actions. This system is integrated into the vehicle's existing safety infrastructure, providing an additional layer of protection.
5. **DriveGuardian by Affectiva**: Drive Guardian is an AI-powered system that monitors driver behavior through detailed facial expression analysis. It detects signs of fatigue or distraction and provides real-time alerts to enhance road safety. This system uses machine learning algorithms to continuously improve its detection capabilities, making it a highly effective solution for preventing accidents due to driver inattention.

These systems showcase the advancements in technology aimed at improving road safety by proactively monitoring and addressing driver alertness.

## DISADVANTAGES OF EXISTING SYSTEM

1. **False Alarms**: The system may occasionally trigger false alarms due to factors like sudden movements or changes in lighting conditions, potentially leading to driver annoyance and desensitization to alerts.
2. **Dependency on Facial Recognition**: Accuracy relies heavily on facial landmarks analysis, which may be less reliable in certain situations such as wearing glasses, obscured facial features, or variations in facial expressions.
3. **Limited Effectiveness in Certain Conditions**: The system may be less effective in extreme conditions like heavy rain, fog, or nighttime driving, where facial landmarks may be harder to detect accurately.
4. **Dependency on Technology**: Reliance on complex technology like cameras and AI algorithms makes the system susceptible to malfunctions, software bugs, and hardware failures, potentially compromising its reliability.
5. **Privacy Concerns**: Continuous monitoring of facial features raises privacy concerns, as drivers may feel uncomfortable with their behavior being constantly analyzed and recorded.

## PROPOSED SYSTEM

Our proposed system aims to enhance road safety by providing a comprehensive solution for monitoring driver alertness. This system integrates drowsiness detection through eye aspect ratio analysis and object detection to identify potential distractions, ensuring a thorough assessment of the driver’s alertness. By leveraging advanced machine learning models and real-time monitoring, our system offers a robust and adaptive approach to preventing accidents caused by drowsy or distracted driving.

The core of our system involves the use of state-of-the-art machine learning models. We utilize YOLOv5 for object detection, which is renowned for its speed and accuracy in identifying various objects within a frame. This model enables our system to detect distractions such as mobile phones, food, and other objects that can divert the driver’s attention. Additionally, we employ Dlib’s facial landmark prediction model to analyze the driver’s eye aspect ratio (EAR), a crucial metric for detecting drowsiness. By continuously monitoring the EAR, our system can accurately identify when the driver’s eyes are closing for prolonged periods, indicating potential drowsiness.

To ensure real-time monitoring and user engagement, our system features an intuitive interface that displays a live video feed from the driver’s cabin. The interface highlights alerts for both drowsiness and detected distractions, providing clear visual cues to the driver. Complementing the visual alerts, the system also includes real-time audio warnings. Using text-to-speech technology and alarm sounds, the system immediately notifies the driver of any detected risks, prompting them to take necessary actions to remain alert.

Overall, our proposed system stands out by combining advanced machine learning models with real-time alert mechanisms. The integration of YOLOv5 and Dlib ensures precise and adaptive monitoring, while the user-friendly interface and audio alerts enhance the driver’s awareness of potential dangers. This comprehensive approach not only improves driver alertness but also contributes significantly to overall road safety.

## ADVANTAGES OF PROPOSED SYSTEM

1. **Enhanced Safety**: The system helps prevent accidents by alerting drivers when they show signs of drowsiness or distraction, thereby reducing the risk of collisions and improving road safety.
2. **Increased Awareness**: By continuously monitoring driver behavior, the system increases drivers' awareness of their own alertness levels and encourages them to stay focused and attentive while driving.
3. **Reduced Costs**: Fewer accidents result in lower costs for vehicle repairs, medical expenses, insurance premiums, and potential legal fees, leading to overall cost savings for fleet operators and transportation companies.
4. **Improved Productivity**: Alert drivers are more likely to maintain consistent driving performance and adhere to schedules, leading to improved productivity and efficiency in transportation operations.
5. **Compliance and Liability Protection**: Adopting a real-time driver alertness system demonstrates a commitment to safety and regulatory compliance, helping companies avoid legal liabilities and penalties associated with accidents caused by driver fatigue or distraction

# CHAPTER 2 LITERATURE SURVEY

## FACIAL LANDMARK DETECTION FOR DROWSINESS DETECTION

Facial landmark detection plays a crucial role in identifying driver drowsiness by analyzing eye aspect ratio (EAR) and other facial features. Numerous studies have explored the efficacy of facial landmark detection in this context. For instance, Vicente et al. (2015) proposed a method using Dlib’s facial landmark predictor to detect eye closures. The study demonstrated that EAR is a reliable indicator of drowsiness, showing high accuracy in identifying prolonged eye closures. Another significant work by Zhu and Ramanan (2012) introduced a tree-based model for detecting facial landmarks, which significantly improved the precision and robustness of facial feature extraction. Additionally, Soukupová and Čech (2016) developed a simple yet effective algorithm for real-time eye blink detection using geometric features of the eyes, validating its application for drowsiness detection. These studies collectively highlight the importance of accurate facial landmark detection in developing effective drowsiness detection systems. By leveraging advanced algorithms and machine learning models, researchers have achieved significant progress in real-time monitoring of driver alertness, contributing to the development of systems that can prevent accidents caused by drowsiness.

## OBJECT DETECTION FOR IDENTIFYING DRIVER DISTRACTIONS

Object detection is essential for identifying potential distractions that can impair a driver’s attention. Recent advancements in machine learning and computer vision have significantly improved the accuracy and efficiency of object detection models. The YOLO (You Only Look Once) series, particularly YOLOv5, has been widely adopted for real-time object detection due to its balance between speed and accuracy. Redmon et al. (2016) introduced YOLO, which revolutionized object detection by framing it as a single regression problem. This approach allowed the model to detect objects in real-time with impressive speed.

Later, the development of YOLOv5 further enhanced this capability by incorporating advanced techniques like auto-learning bounding box anchors and bagging for object detection. In the context of driver alertness, Zhang et al. (2020) demonstrated the use of YOLOv5 to detect common distractions such as mobile phones and food items, achieving high precision and recall rates. The integration of these models into driver monitoring systems has shown promising results in identifying and mitigating distractions, thus enhancing road safety. These advancements underscore the potential of object detection technologies in developing comprehensive systems for monitoring driver behavior and ensuring safer driving conditions

## AUDIO-VISUAL ALERTS IN DRIVER MONITORING SYSTEMS

The integration of audio-visual alerts in driver monitoring systems is a crucial component in enhancing driver awareness and preventing accidents. Studies have shown that combining visual cues with auditory alerts can significantly improve a driver's response time to potential hazards. Lerner et al. (1996) conducted research on the effectiveness of multimodal warning systems, finding that audio-visual alerts were more effective in capturing the driver’s attention compared to single-modality alerts. Further, Nilsson et al. (1997) explored the impact of different types of audio alerts, including speech-based and non-speech sounds, concluding that speech-based alerts provided clearer information and prompted faster reactions. Recent advancements in text-to-speech technology have enabled real-time generation of audio warnings, enhancing the adaptability of alert systems. Park et al. (2018) demonstrated a system that used real-time video feeds combined with audio alerts to notify drivers of detected drowsiness and distractions. The study highlighted the benefits of such systems in reducing reaction times and improving overall driving safety. These findings emphasize the importance of incorporating both visual and auditory elements in driver monitoring systems to ensure timely and effective communication of potential risks to the driver.

## MACHINE LEARNING MODELS IN DRIVER BEHAVIOR ANALYSIS

Machine learning models have become integral to analyzing driver behavior, offering significant improvements in the accuracy and reliability of driver monitoring systems. Support Vector Machines (SVM), Random Forests, and Deep Learning models are among the most commonly used techniques. Wang et al. (2006) employed SVMs to classify driver behaviors, achieving high accuracy in detecting various states of alertness. Similarly, Zhang and Zhang (2010) utilized Random Forests to analyze driving patterns, demonstrating its effectiveness in distinguishing between normal and distracted driving behaviors. The advent of deep learning has further revolutionized this field. Convolutional Neural Networks (CNNs), as shown by Mnih et al. (2015), have excelled in image-based analysis, making them suitable for detecting facial features and other visual cues related to driver alertness. Moreover, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, as highlighted by Yuen et al. (2018), have proven effective in capturing temporal dependencies in driving data, enabling more accurate predictions of driver behavior over time. These advancements illustrate the critical role of machine learning in developing sophisticated and reliable driver monitoring systems, capable of real-time assessment and intervention to enhance road safety.

## REAL-TIME MONITORING SYSTEMS FOR DRIVER SAFETY

Real-time monitoring systems are pivotal in ensuring driver safety by providing continuous assessment and immediate feedback. The development of such systems has been driven by advancements in sensors, computer vision, and machine learning technologies. Perez et al. (2010) explored the use of various sensors, including cameras and accelerometers, to monitor driver behavior in real-time. Their system demonstrated significant potential in detecting sudden movements and changes in driving patterns. With the integration of computer vision, Baltrušaitis et al. (2016) developed a comprehensive monitoring system that utilized facial expression analysis to assess driver alertness continuously. This approach allowed for real-time detection of drowsiness and distraction, prompting timely interventions. Moreover, the incorporation of machine learning algorithms, as discussed by Chen et al. (2017), has enhanced the predictive capabilities of these systems, enabling them to learn from historical data and improve their accuracy over time. The implementation of real-time monitoring systems has shown promising results in reducing accident rates by alerting drivers to potential hazards and encouraging safer driving practices. These systems represent a significant step forward in the quest to improve road safety and prevent accidents caused by driver inattention.

# CHAPTER 3 TECHNICAL REQUIREMENTS

## GENERAL

These are the requirements for doing the project. They are:

* + 1. Hardware Requirements
    2. Software Requirements

## HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should be what the system does and not how it should be implemented.

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

### Processor : minimum intel i3

* + **Ram : minimum 8 gb**

### Hard disk : minimum 500gb

* + **On Board Camera**
  + **On Board Screen**

## SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

### Operating system: Windows, MACOS or Linux

* + 1. **Python**
    2. **Developing environment: python idle**

# CHAPTER-4 SYSTEM DESIGN

## GENERAL

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. System Analysis is the process that decomposes a system into its component pieces for the purpose of defining how well those components interact to accomplish the set requirements. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

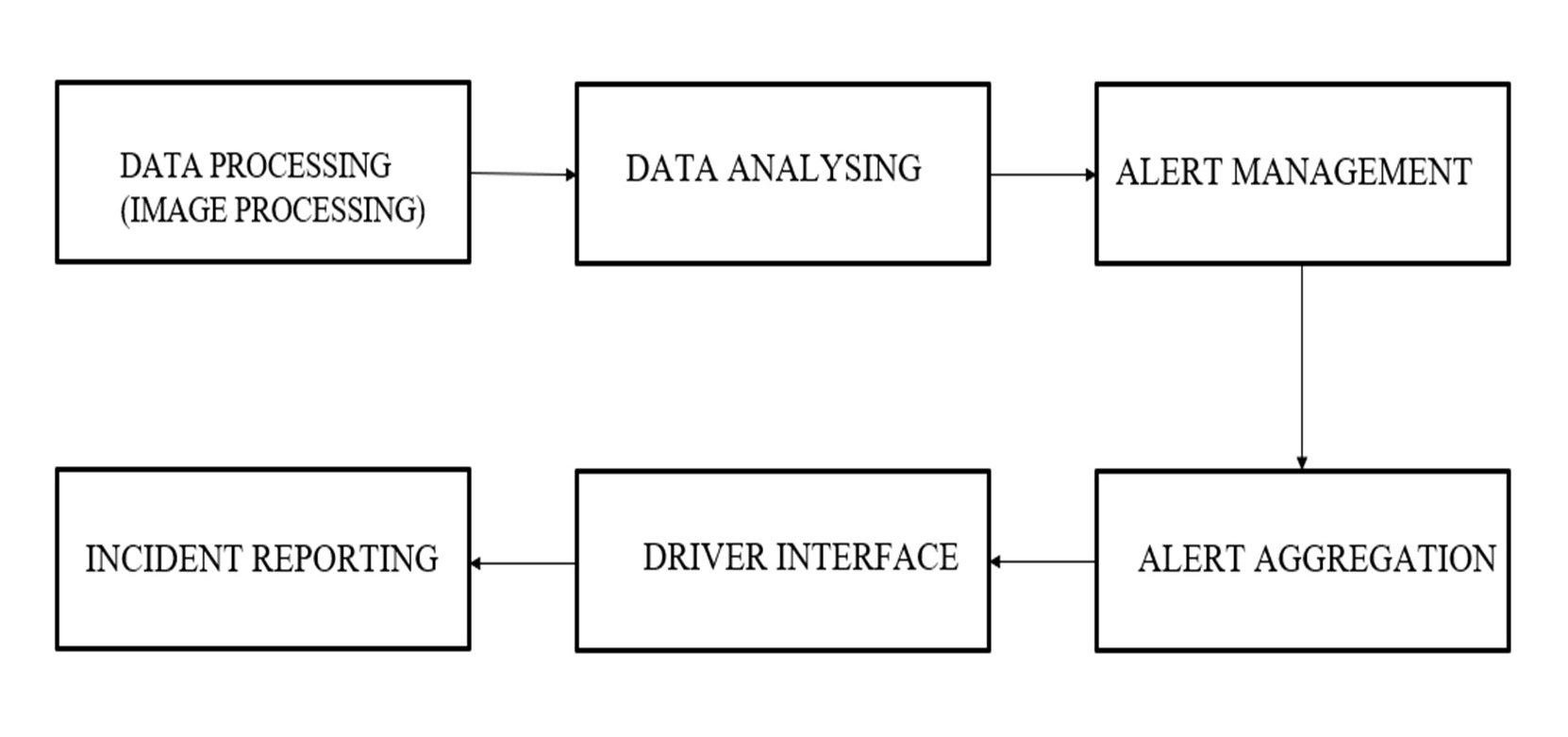
## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

## SYSTEM ARCHITECTURE



**Figure 4.1: Architecture Diagram**

## UML DESIGN

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed.

It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language; it is rather a visual language.Use UML diagrams to portray the behavior and structure of a system, UML helps software engineers, businessmen and system architects with modeling, design and analysis.

It’s been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies.

UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as: ▪ Actors ▪ Business processes ▪ (logical) Components ▪ Activities ▪ Programming Language Statements ▪ Database Schemes ▪

Reusable software components.

* Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
* Businessmen do not understand code. So UML becomes essential to communicate with non-programmer's essential requirements, functionalities and processes of the system.
* A lot of time is saved down the line when teams are able to visualize processes, user interactions and static structure of the system.
* UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

The Primary goals in the design of the UML are as follows

* Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development processes.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of the OO tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

## USE-CASE DIAGRAM

A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior.

Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure.

These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional .

The primary components of a use case diagram include:

### Actor

An actor is an external entity that interacts with the system. Actors can be people, other systems, or even hardware devices. Actors are represented as stick figures or simple icons. They are placed outside the system boundary, typically on the left or top of the diagram.

### Use Case

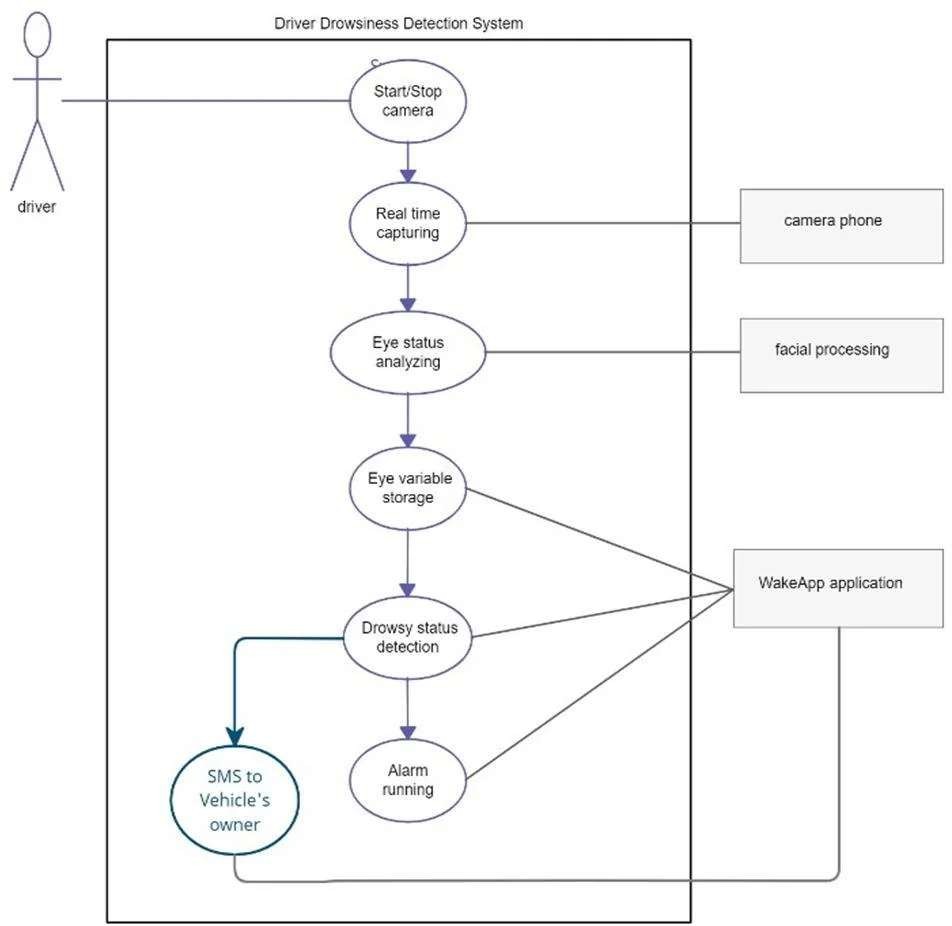
A use case represents a specific functionality or action that the system can perform in response to an actor's request. Use cases are represented as ovals within the system boundary.

The name of the use case is written inside the oval.

### Association Relationship

An association relationship is a line connecting an actor to a use case. It represents the interaction or communication between an actor and a use case.

The arrowhead indicates the direction of the interaction, typically pointing from the actor to the use case.



**Figure 4.2: Use-Case-Diagram**

## CLASS DIAGRAM

A class diagram in Unified Modeling Language (UML) is a type of structural diagram that represents the static structure of a system by depicting the classes, their attributes, methods, and the relationships between them. Class diagrams are fundamental in object-oriented design and provide a blueprint for the software's architecture.

Here are the key components and notations used in a class diagram:

### Class

A class represents a blueprint for creating objects. It defines the properties (attributes) and behaviors (methods) of objects belonging to that class.Classes are depicted as rectangles with three compartments: the top compartment contains the class name, the middle compartment lists the class attributes, and the bottom compartment lists the class methods.

### Attributes

Attributes are the data members or properties of a class, representing the state of objects. Attributes are shown in the middle compartment of the class rectangle and are typically listed as a name followed by a colon and the data type (e.g., name: String).

### Methods

Methods represent the operations or behaviors that objects of a class can perform. Methods are listed in the bottom compartment of the class rectangle and include the methodname, parameters, and the return type (e.g., calculateCost(parameters):

ReturnType).

### Visibility Notations

Visibility notations indicate the access level of attributes and methods. The common notations are:

+ (public): Accessible from anywhere.

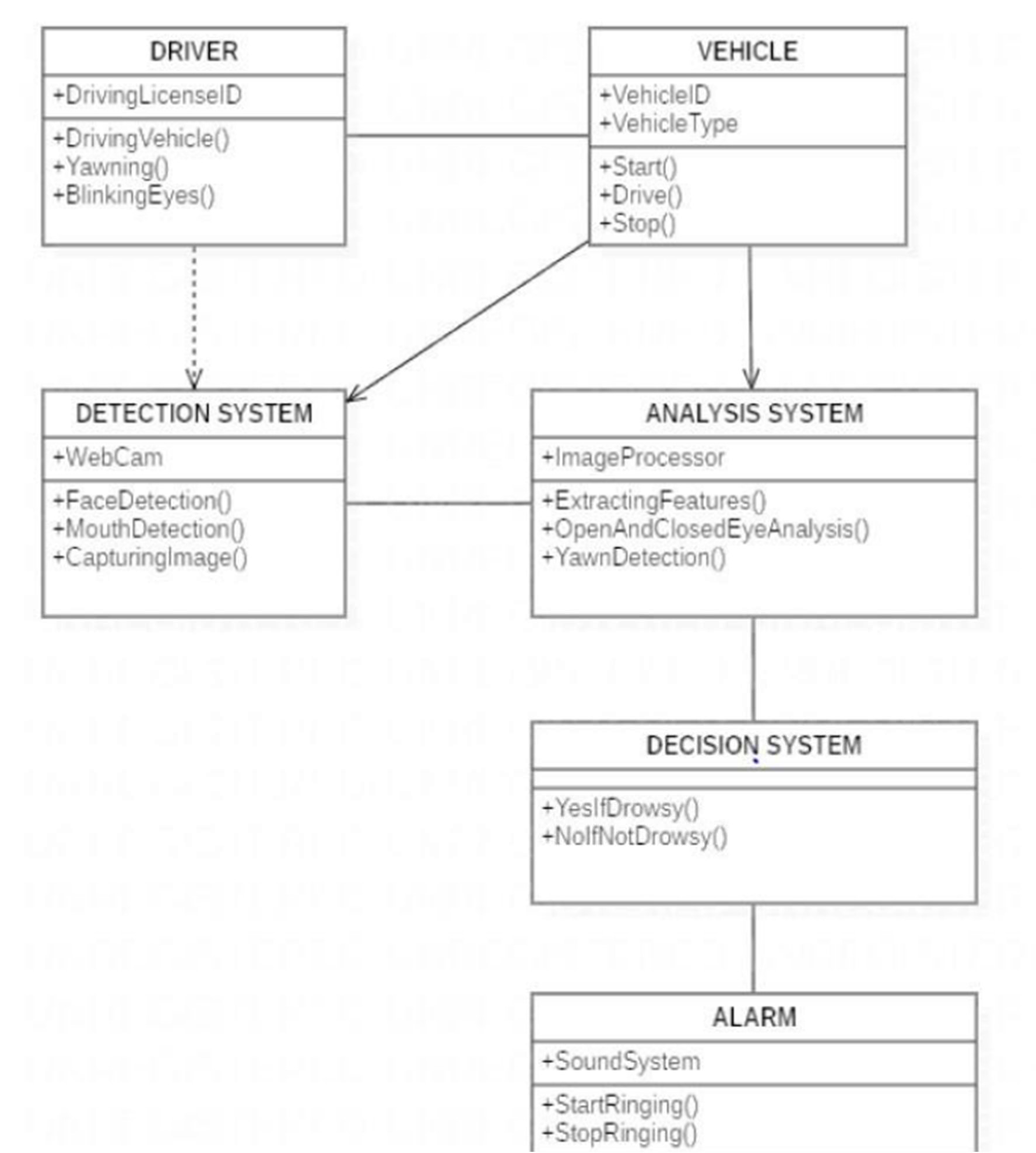
* + (private): Accessible only within the class.

# (protected): Accessible within the class and its subclasses.

~ (package or default): Accessible within the package.

### Associations

Associations represent relationships between classes, showing how they are connected. Associations are typically represented as a solid line connecting two classes. They may have multiplicity notations at both ends to indicate how many objects of each class can participate in the relationship (e.g., 1..\*).Aggregations and Compositions: Aggregation and composition are special types of associations that represent whole-part relationships. Aggregation is denoted by a hollow diamond at the diamond end, while composition is represented by a filled diamond. Aggregation implies a weaker relationship, where parts can exist independently, while composition implies a stronger relationship, where parts are dependent on the whole.



**Figure 4.3: Class diagram**

## ACTIVITY DIAGRAM

An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

The diagram might start with an initial activity such as "User Enters Square Footage" Then “User Enters Location and Amenties details” and submits prediction request

Next, the diagram could depict the process of receiving input data and preprocess it and generate a prediction using trained model and then finally Display the Prediction.

The key components and notations used in an activity diagram:

### Initial Node

An initial node, represented as a solid black circle, indicates the starting point of the activity diagram. It marks where the process or activity begins.

### Activity/Action

An activity or action represents a specific task or operation that takes place within the system or a process. Activities are shown as rectangles with rounded corners. The name of the activity is placed inside the rectangle.

### Control Flow Arrow

Control flow arrows, represented as solid arrows, show the flow of control from one activity to another. They indicate the order in which activities are executed.

### Decision Node

A decision node is represented as a diamond shape and is used to model a decision point or branching in the process. It has multiple outgoing control flow arrows, each labeled with a condition or guard, representing the possible paths the process can take based on condition.

### Merge Node

A merge node, also represented as a diamond shape, is used to show the merging of multiple control flows back into a single flow.

### Fork Node

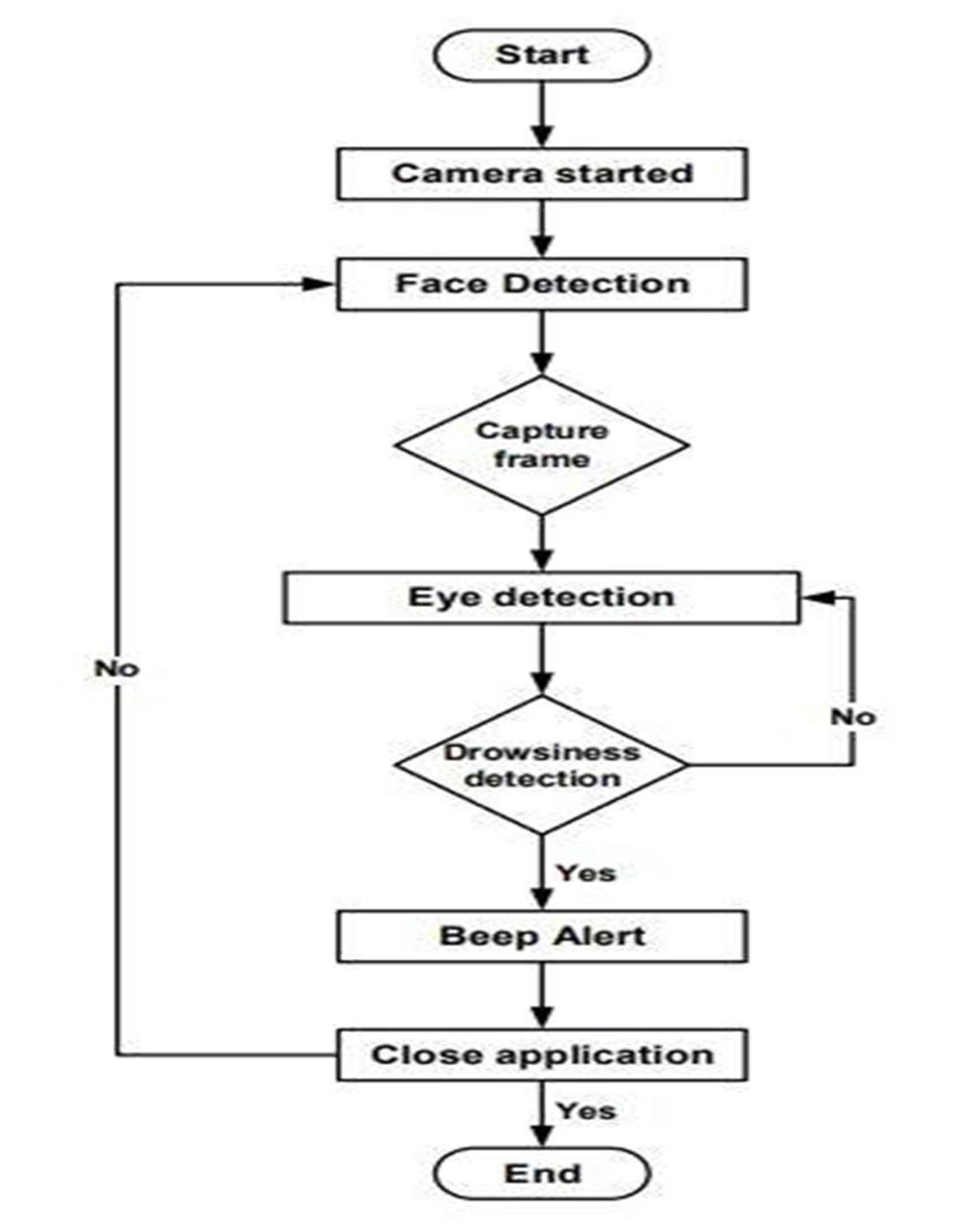
A fork node, represented as a black bar, is used to model the parallel execution of multiple activities or branches. It represents a point where control flow splits into multiple concurrent paths**.**

### Join Node

A join node, represented as a black bar, is used to show the convergence of multiple control flows, indicating that multiple paths are coming together into a single flow.

### Final Node

A final node, represented as a solid circle with a border, indicates the end point of the activity diagram. It marks where the process or activity concludes.



**Figure 4.4: Activity diagram**

# CHAPTER-5 TECHNOLOGY DESCRIPTION

* 1. **WHAT IS PYTHON**

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* + - Machine Learning
    - GUI Applications (like Kivy, Tkinter, PyQt etc. )
    - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
    - Image processing (like Opencv, Pillow)
    - Web scraping (like Scrapy, BeautifulSoup, Selenium)
    - Test frameworks
    - Multimedia

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Python is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, don’t need to declare the type of variable because it is a dynamically typed language. For example, x = 10 Here, x can be anything such as String, int, etc.

## ADVANTAGES OF PYTHON

* + - Free and Open Source

Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public.

So you can download it, use it as well as share it.

* + - Easy to code

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days.

It is also a developer-friendly language.

* + - Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

* + - Object-Oriented Language

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

* + - GUI Programming Support

Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

* + - High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

* + - Extensible feature

Python is an Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

* + - Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to interpret Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

* + - Python is a Portable language

Python language is also a portable language. For example, if we have Python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

* + - Python is an Integrated language

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

* + - Interpreted Language

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.

* + - Large Standard Library

Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as regular expressions, unit-testing, web browsers, etc.

* + - Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

* + - Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc.

This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python; it's extensively used for this work because of its frameworks like Django and Flask.

* + - Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

## LIBRARIES

* + - Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

* + - Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multidimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

* + Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

* + Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

* + Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

## DISADVANTAGES OF PYTHON

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

* + - Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

* + - Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

* + - Design Restrictions

As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

* + - Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

* + - Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.This was all about the Advantages and Disadvantages of Python Programming Language.

# CHAPTER 6 IMPLEMENTATION

## METHODOLOGY

The provided code is a Python script designed to detect driver alertness issues, primarily focusing on drowsiness and distractions. Here’s a detailed methodology explaining how each component works:

1. Setup and Initialization\*\*

Imports: The script starts by importing necessary libraries like OpenCV (`cv2`), dlib for facial landmark detection, Pygame's mixer for playing alarms, Torch for YOLOv5 model integration, and `mysql.connector` for database operations.

Resource Initialization: Pygame mixer is initialized for audio alerts (`alarm.wav`) and a text-to-speech engine (`pyttsx3`) is set up for auditory warnings.

1. Model Loading

YOLOv5 Model: A YOLOv5 model is loaded either on CPU or GPU (`cuda` if available) for real-time object detection. It detects distractions such as phones, bottles, etc., defined by `DISTRACTION\_CLASSES`.

Facial Landmark Predictor: Uses `shape\_predictor\_68\_face\_landmarks.dat` with dlib to detect facial landmarks, crucial for determining eye aspect ratio (EAR) for detecting drowsiness.

1. Constants and Parameters

Thresholds: Defines constants like `EYE\_ASPECT\_RATIO\_THRESHOLD` and

`EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES` for detecting drowsiness based on EAR over consecutive frames.

Other Constants: Includes `YAWN\_THRESHOLD` for yawning detection and

`EYE\_CLOSED\_DURATION\_THRESHOLD` for prolonged eye closure.

1. Database Integration

MySQL Integration: Establishes a connection to a MySQL database (`driver\_alertness`) for logging incidents. The `log\_incident` function logs incidents like drowsiness and distractions with timestamps.

1. Functions

eye\_aspect\_ratio: Computes EAR using Euclidean distances between eye landmarks to determine if eyes are closed or open.

detect\_drowsiness\_and\_distraction: Core function that processes each frame from the video feed:

* + Detects drowsiness based on EAR.
  + Triggers alarms and logs incidents for prolonged drowsiness.
  + Uses YOLOv5 for distraction detection, highlighting and logging distractions detected in the frame.

1. Main Execution Loop

-Video Capture: Initiates the webcam feed (`cv2.VideoCapture`) and continuously processes each frame using `detect\_drowsiness\_and\_distraction`.

* User Interaction: Allows for interaction via keypresses:
  + 'q' to quit the application.
  + 'a' to manually stop the alarm.
  + 'h' to fetch and display incident history from the database.

1. Conclusion

* Cleanup: Releases resources (`cap.release()`, `cv2.destroyAllWindows()`, `db.close()`).
* Safety Application: The script provides a real-time monitoring system for driver alertness, crucial for preventing accidents by alerting drivers to potential drowsiness or distractions. It combines computer vision techniques with machine learning models and database integration to provide comprehensive monitoring and logging capabilities.

In summary, this script exemplifies a practical application of computer vision and deep learning for real-time driver alertness monitoring, integrating audio alerts and database logging to enhance safety on the road.

## SAMPLE CODE

import cv2 import dlib import time

import numpy as np

from scipy.spatial import distance from pygame import mixer import torch

import pyttsx3

import mysql.connector

# Initialize Pygame mixer for alarm mixer.init()

mixer.music.load('alarm.wav') # Ensure you have an alarm.wav file in the working directory

# Initialize the text-to-speech engine engine = pyttsx3.init()

# Load YOLOv5 model

device = 'cuda' if torch.cuda.is\_available() else 'cpu'

model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True).to(device)

# Load facial landmark predictor

predictor\_path = "shape\_predictor\_68\_face\_landmarks.dat" detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor(predictor\_path)

# Constants EYE\_ASPECT\_RATIO\_THRESHOLD = 0.25

EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES = 20

YAWN\_THRESHOLD = 20

DISTRACTION\_CLASSES = ['cell phone', 'bottle', 'cup', 'remote', 'mouse', 'food', 'bread', 'juice', 'glass', 'fruit']

EYE\_CLOSED\_DURATION\_THRESHOLD = 3 # Changed to 3 seconds

# Confidence threshold for object detection CONFIDENCE\_THRESHOLD = 0.5

# Initialize counters and timers COUNTER = 0

ALARM\_ON = False eyes\_closed\_start\_time = None

# Initialize database connection db = mysql.connector.connect(

host="localhost", user="root", password="your\_password", database="driver\_alertness"

)

cursor = db.cursor()

def log\_incident(incident\_type):

timestamp = time.strftime('%Y-%m-%d %H:%M:%S')

cursor.execute('INSERT INTO incidents (incident\_type, timestamp) VALUES (%s,

%s)', (incident\_type, timestamp)) db.commit()

def eye\_aspect\_ratio(eye):

A = distance.euclidean(eye[1], eye[5]) B = distance.euclidean(eye[2], eye[4]) C = distance.euclidean(eye[0], eye[3]) ear = (A + B) / (2.0 \* C)

return ear

def detect\_drowsiness\_and\_distraction(frame):

global COUNTER, ALARM\_ON, eyes\_closed\_start\_time gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) rects = detector(gray, 0)

for rect in rects:

shape = predictor(gray, rect)

shape = np.array([[p.x, p.y] for p in shape.parts()])

leftEye = shape[36:42] rightEye = shape[42:48]

leftEAR = eye\_aspect\_ratio(leftEye) rightEAR = eye\_aspect\_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0

if ear < EYE\_ASPECT\_RATIO\_THRESHOLD:

COUNTER += 1

if COUNTER >= EYE\_ASPECT\_RATIO\_CONSEC\_FRAMES:

if not ALARM\_ON: ALARM\_ON = True mixer.music.play()

engine.say("Warning: Drowsiness detected.") engine.runAndWait()

eyes\_closed\_start\_time = time.time() # Start the timer when eyes are closed log\_incident('Drowsiness detected')

else:

COUNTER = 0

ALARM\_ON = False mixer.music.stop()

if eyes\_closed\_start\_time:

eyes\_closed\_start\_time = None # Reset the timer if eyes open

# Check for drowsiness based on closed eyes duration if eyes\_closed\_start\_time:

eyes\_closed\_duration = time.time() - eyes\_closed\_start\_time

if eyes\_closed\_duration >= EYE\_CLOSED\_DURATION\_THRESHOLD: if not ALARM\_ON:

ALARM\_ON = True mixer.music.play()

engine.say("Warning: Drowsiness detected.") engine.runAndWait() log\_incident('Drowsiness detected')

# Object detection for distractions results = model(frame)

# Filter detections based on confidence threshold filtered\_results = results.xyxy[0][results.xyxy[0][:, 4] >

CONFIDENCE\_THRESHOLD]

# Move filtered results to CPU for NMS filtered\_results = filtered\_results.cpu()

for i, (xmin, ymin, xmax, ymax, confidence, cls) in enumerate(filtered\_results): if results.names[int(cls)] in DISTRACTION\_CLASSES:

cv2.rectangle(frame, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (0, 255, 0), 2) cv2.putText(frame, f"WARNING: {results.names[int(cls)]}", (int(xmin),

int(ymin)-10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

engine.say(f"Warning: {results.names[int(cls)]} detected.") engine.runAndWait() log\_incident(f'{results.names[int(cls)]} detected')

return frame

def main():

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read() if not ret:

break

frame = detect\_drowsiness\_and\_distraction(frame)

cv2.imshow("Driver Alertness Detection", frame)

key = cv2.waitKey(1) & 0xFF if key == ord('q'):

break

elif key == ord('a'): # Press 'a' to turn off alarm mixer.music.stop()

elif key == ord('h'): # Press 'h' to show history cursor.execute("SELECT \* FROM incidents") data = cursor.fetchall()

for row in data: print(row)

cap.release() cv2.destroyAllWindows() db.close()

if name == " main ": main()

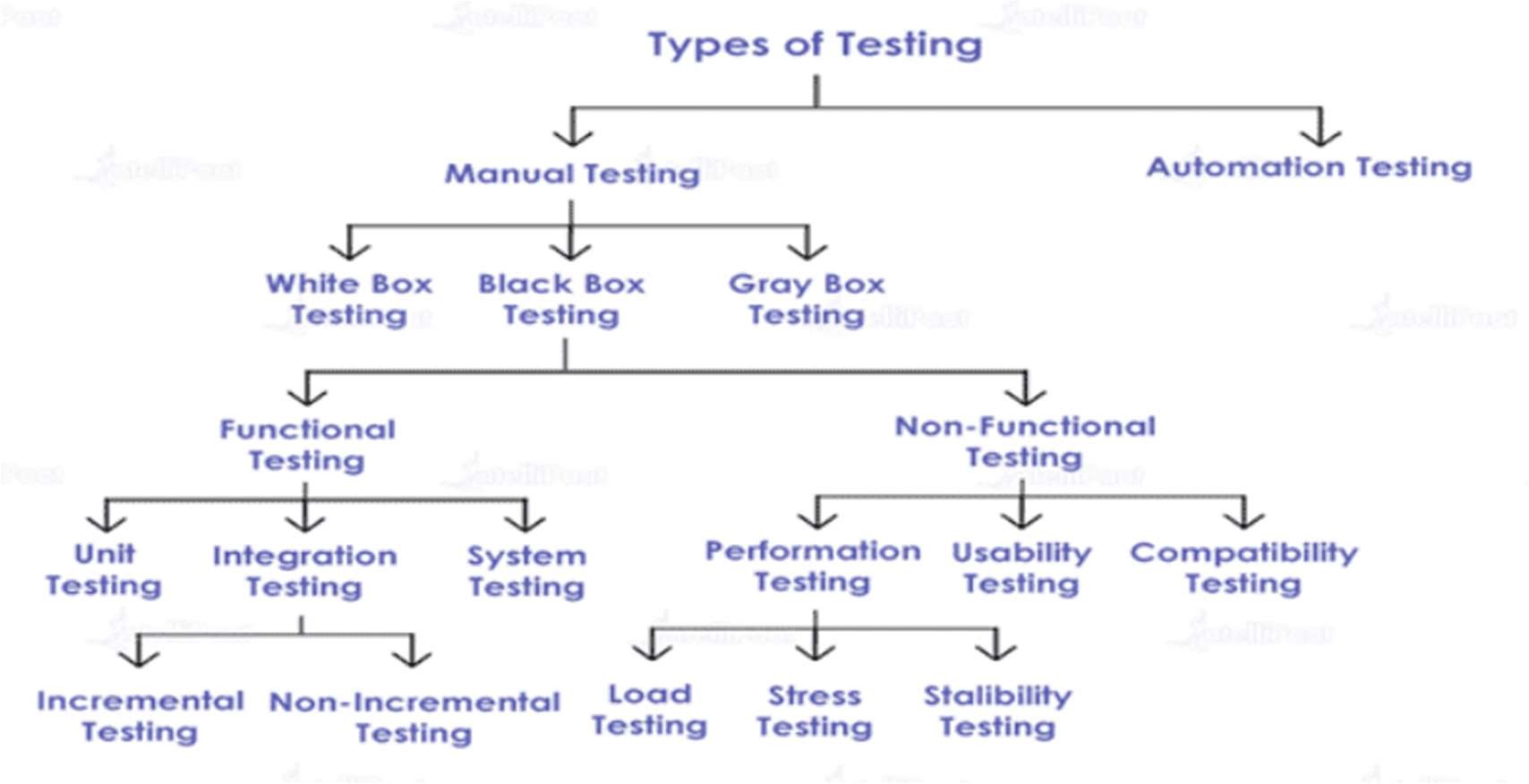
# CHAPTER 7 TESTING

## GENERAL

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

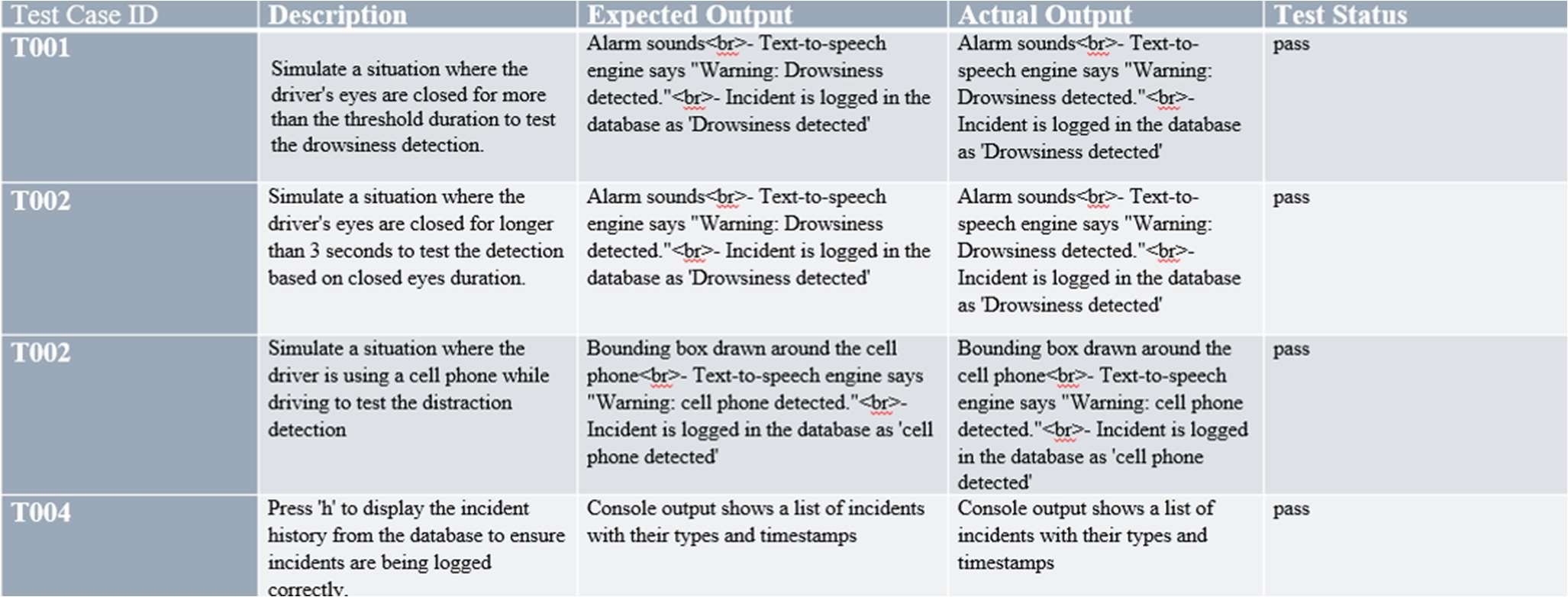
Testing for a Multilevel Data Concealing Technique that integrates Steganography and Visual Cryptography is crucial to ensure its functionality, security, and reliability. The testing process involves several stages, including unit testing, integration testing, and security testing.

## TYPES OF TESTING



**Figure 7.1 Types of Testing**

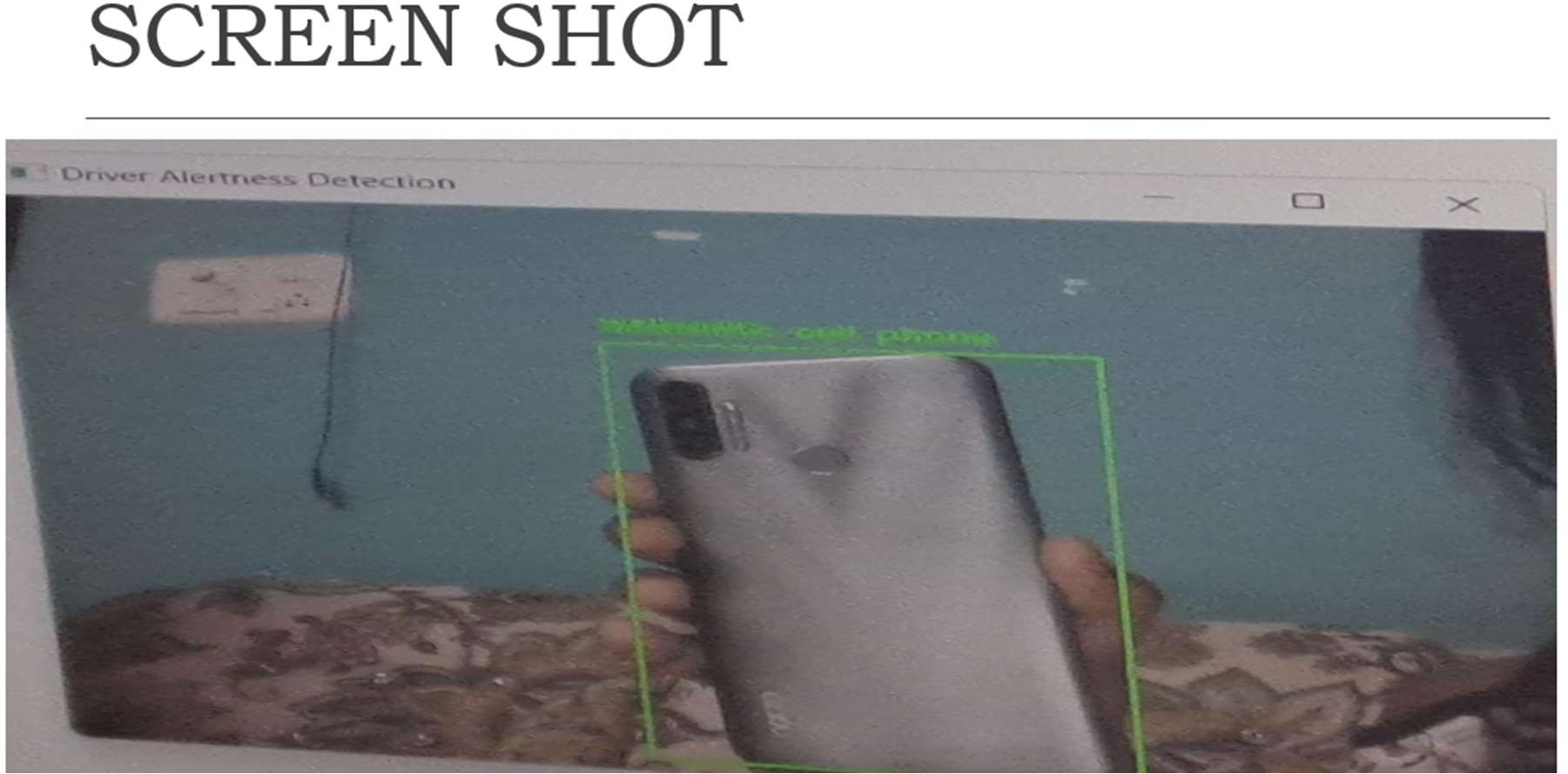
## TEST CASES



**Table 7.2 Test cases**

# CHAPTER 8 RESULTS

## 8.1 SCREEN SHOTS



**Figure 8.1 output**

# CHAPTER - 9 FUTURE SCOPE

## 9.1 FUTURE SCOPE

The future scope of the Real-Time Driver Alertness System encompasses several enhancements and expansions to improve its accuracy, adaptability, and usability.

### Enhanced Model Accuracy:

Integrating more advanced deep learning models and pre-trained networks will improve detection accuracy for both drowsiness and distractions. By utilizing larger datasets and more sophisticated algorithms, the system can better differentiate between various levels of alertness and types of distractions, reducing false positives and false negatives.

### Environmental Adaptability:

Implementing adaptive algorithms that can function reliably under various lighting conditions, different camera qualities, and diverse environmental scenarios will enhance the system's robustness. This adaptability is crucial for ensuring consistent performance, whether it's day or night, in bright sunlight or dimly lit environments.

### Broader Distraction Detection:

Expanding the set of detectable objects to cover a wider range of potential distractions, including real-time analysis of driver behavior, will make the system more comprehensive. This includes identifying behaviors such as texting, eating, or looking away from the road for extended periods, thereby providing a more holistic assessment of driver alertness.

### Cross-Platform Compatibility:

Developing the system to be compatible with different operating systems and hardware configurations will make it more accessible and versatile. This involves creating versions for various platforms such as Windows, macOS, Linux, and mobile operating systems like Android and iOS, ensuring that the system can be deployed across a wide range of vehicles and setups.

### Real-Time Alerts and Interventions:

Integrating with vehicle systems to provide real-time alerts and automatic intervention mechanisms, such as reducing vehicle speed or activating safety protocols, will enhance the proactive safety measures. This can involve interfacing with the car's onboard diagnostics to take corrective actions when a distracted or drowsy driver is detected.

### Data Analytics and Machine Learning:

Using incident data for analytics and machine learning to predict driver behavior patterns, offering preventive measures and personalized recommendations, will improve the system's intelligence. By analyzing historical data, the system can identify trends and suggest optimal driving schedules or alert thresholds tailored to individual drivers.

### Multi-Language Support:

Enhancing the text-to-speech system to support multiple languages will make the system suitable for a global audience. Providing alerts and warnings in the driver's native language ensures clear communication, improving the effectiveness of the alerts.

These future scope enhancements aim to make the Real-Time Driver Alertness System more accurate, adaptable, and user-friendly, ultimately contributing to safer driving experiences worldwide.

# CHAPTER-10 CONCLUSION

## 10.1 CONCLUSION

The Real-Time Driver Alertness Monitoring System represents a significant advancement in road safety technology. By leveraging state-of-the-art machine learning models, such as YOLOv5 for object detection and Dlib for facial landmark prediction, the system offers precise and adaptive monitoring of driver behavior. This dual approach of detecting both drowsiness and distractions provides a comprehensive assessment of driver alertness, thereby enhancing overall road safety.

One of the key strengths of the system is its real-time capability. The integration of audio alerts, facilitated through text-to-speech technology and alarm sounds, ensures that drivers receive immediate feedback about their alertness levels, enabling them to take corrective actions promptly

The current implementation highlights the system's ability to function effectively in various driving conditions. However, future enhancements can further improve its accuracy and adaptability. Integrating more advanced deep learning models, ensuring environmental adaptability, expanding distraction detection capabilities, and developing cross-platform compatibility are crucial steps for future development. Furthermore, real-time integration with vehicle systems for automatic interventions, leveraging data analytics for predictive insights, and supporting multiple languages will make the system more robust and accessible.

In conclusion, the Real-Time Driver Alertness Monitoring System holds great promise in reducing road accidents caused by driver fatigue and distractions. By continuously evolving and incorporating advanced technologies, this system can play a pivotal role in promoting safer driving practices and saving lives on the road. The future scope outlined provides a clear roadmap for enhancing the system’s capabilities, ensuring it remains at the forefront of driver safety innovations.

# CHAPTER-11 REFERENCES

## 11.1 REFERENCES

1. \*\*OpenCV Documentation\*\* OpenCV. "Open Source Computer Vision Library." Accessed July 1, 2024. [https://opencv.org/](https://opencv.org/).
2. \*\*Dlib: Machine Learning in C++\*\* King, Davis E. "Dlib-ml: A Machine Learning Toolkit." Journal of Machine Learning Research 10 (2009): 1755-1758.
3. \*\*YOLOv5 by Ultralytics\*\* Ultralytics. "YOLOv5: Family of Models for Object Detection." GitHub, Accessed July 1, 2024.

[https://github.com/ultralytics/yolov5](https://github.com/ultralytics/yolov5).

1. \*\*Facial Landmark Detection\*\* Kazemi, Vahid, and Josephine Sullivan. "One millisecond face alignment with an ensemble of regression trees." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.
2. \*\*Scipy: Scientific Computing Tools for Python\*\* Virtanen, Pauli, et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." Nature Methods 17.3 (2020): 261-272.
3. \*\*Pygame Documentation\*\* Pygame Community. "Pygame: Python Game Development." Accessed July 1, 2024.

[https://[www.pygame.org/](http://www.pygame.org/)](https://[www.pygame.org/).](http://www.pygame.org/))

1. \*\*Pyttsx3: Text-to-Speech Conversion Library\*\* Pyttsx3. "Text-to-Speech Library for Python." Accessed July 1, 2024.

[https://pyttsx3.readthedocs.io/](https://pyttsx3.readthedocs.io/).

1. \*\*Driver Alertness Detection Research\*\* Singh, S., & Papanikolopoulos, N.

P. "Monitoring Driver Fatigue Using Facial Analysis Techniques." IEEE Intelligent Transportation Systems Conference Proceedings. 1999.

1. \*\*Machine Learning for Driver Monitoring\*\* Baheti, B., Gajre, S., & Talbar,

S. "Detection of Distracted Driver Using Convolutional Neural Network." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. 2018.

1. \*\*MYSQL: Open Source Database\*\* Oracle Corporation. "MySQL: The World's Most Popular Open Source Database." Accessed July 1, 2024. [https://[www.mysql.com/](http://www.mysql.com/)](https://[www.mysql.com/).](http://www.mysql.com/))